To appear in the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing

IMPROVING THE CAPACITY OF VERY DEEP NETWORKS WITH MAXOUT UNITS

Oyebade K. Oyedotun, Abd El Rahman Shabayek, Djamila Aouada, Björn Ottersten

Interdisciplinary Centre for Security, Reliability and Trust (SnT),
University of Luxembourg, L-1855, Luxembourg
{oyebade.oyedotun, abdelrahman.shabayek, djamila.aouada, bjorn.ottersten}@uni.lu

ABSTRACT

Deep neural networks inherently have large representational power for approximating complex target functions. However, models based on rectified linear units can suffer reduction in representation capacity due to dead units. Moreover, approximating very deep networks trained with dropout at test time can be more inexact due to the several layers of nonlinearities. To address the aforementioned problems, we propose to learn the activation functions of hidden units for very deep networks via maxout. However, maxout units increase the model parameters, and therefore model may suffer from overfitting; we alleviate this problem by employing elastic net regularization. In this paper, we propose very deep networks with maxout units and elastic net regularization and show that the features learned are quite linearly separable. We perform extensive experiments and reach state-of-the-art results on the USPS and MNIST datasets. Particularly, we reach an error rate of 2.19% on the USPS dataset, surpassing the human performance error rate of 2.5% and all previously reported results, including those that employed training data augmentation. On the MNIST dataset, we reach an error rate of 0.36% which is competitive with the state-of-the-art results.

***Index Terms*—** Image classification, deep networks, residual learning, neural networks

1. INTRODUCTION

Deep networks have become very useful for many computer vision applications. Generally, deep learning relies on learning several levels of hierarchical representations for data; this has been shown to encourage the disentanglement of factors of variations in data [1]. In fact, at representation level, deep neural networks have been shown to require fewer parameters as against shallow models for learning some given target functions [2]. In this work, for the sake of clarity, we consider models with more than 10 hidden layers as very deep networks. Some years back, neural network models of 3-5 hidden layers were considered deep [3][4]. However, many works [5][2] have suggested the benefit of depth for learning highly varying functions. In recent times, neural network models of 20-200 layers have been reported in some works [6][7][8]; such models are now referred to as *very deep models* [8]. Training models with more than 15 hidden layers (i.e. very deep networks) can result into difficulty in optimization [7]. For example, [8] performed experiments to show that even very deep networks trained with batch normalization suffer optimization problems as the model depth grows; albeit, batch normalization is helpful when models have less than 20 layers. In fact, going significantly deeper has not been that successful until recently. In [6][7], training difficulty was alleviated by employing network paths for which features undergo no transformation over model layers and therefore dilution. Interestingly, many works on very deep networks [6][7] [9] [8] relied on rectified linear units (ReLUs) for tackling the problem of units saturation and vanishing gradients in order to improve optimization.
Nevertheless, ReLUs can die out during learning, consequently blocking error gradients and learning nothing [10]. As such, dead ReLUs impact the representation capacity of very deep networks which rely on the backpropagation of error gradients through several layers. In addition, approximating model parameters at test time of very deep ReLUs based networks trained with dropout can be quite inexact due to several layers of nonlinearities [11]. Consequently, we propose to modify the learning of very deep networks such that the model size truly reflects on representation capacity and therefore improves performance. For the very deep networks that we consider in this paper, we build on a previous work [8] that employed residual learning with stochastic identity shortcut connections for improving the *implicit regularization* of very deep networks; the model proposed in the work was referred to as a Stochastic Residual Network (S-ResNet). Our contributions in this paper for improving very deep networks trained with dropout are as follows:

1. Preserve the representation capacity of very deep networks due to dead ReLUs by learning the activation functions of units via maxout [11]. Again, we leverage maxout units to improve parameters approximation at test time.

2. Learn features that are quite linearly separable such that

This work was funded by the National Research Fund (FNR), Luxembourg, under the project reference R-AGR-0424-05-D/Björn Ottersten.

a linear SVM can successfully replace the fully connected layers of the model.

3. Validate the proposed approach on experimental data and demonstrate improved results in comparison to [8]. Particularly, we surpass several state-of-the-art results on the USPS [12][13] and MNIST [11][7][14][15] datasets.

The remainder of this paper is organized as follows. Section 2 gives the background on the very deep model that we build on and the problem statement. In Section 3, we provide details of the proposed approach. Section 4 reports our experiments. Section 5 concludes the paper.

2. BACKGROUND AND PROBLEM STATEMENT

2.1. Background: Residual learning with stochastic input shortcut connections

Residual learning has been used for training very deep networks [6], where identity shortcut connections are used for bridging some specified blocks of layers; this allows the learned transformations to stay close to the input data features [16]. In [6], the output of layer l, $H(x)^l$ is of the form

$$H(x)^l = F^l(H(x)^{l-1}) + H(x)^{l-1}, \qquad (1)$$

where $H(x)^{l-1}$ is the output of layer l-1 feeding into a stack with a specified number of hidden layers with output $H(x)^l$; F^l is the transformation learned at layer l; $1 \leq l \leq L$, $H(x)^0$ is the input data x, and L is the depth of the network.
In [8], a modified form of residual learning was proposed; the constructed model was referred to as a Stochastic Residual Network (S-ResNet); see Fig. 1. It employs additional identity shortcut connections for bridging the hidden layers with input data features. Particularly, the additional shortcut connections are stochastically removed during training to emphasize stochasticity during optimization; that is, with such small perturbations in the hidden layers, the model has a lower probability to converge to sharp minima that have been shown to lead to poor model generalization [17]. As against the original residual learning transformation in (1), the stochastic residual learning transformation proposed in [8] is given as

$$H(x)^l = F^l(H(x)^{l-1}) + H(x)^{l-1} + D * x, \qquad (2)$$

where $D \in \{0, 1\}$ and $D \sim Bernoulli(p_s)$ determines that x (shortcut connection from input) is connected to the stack of hidden layers l with probability p_s; that is, $P(D = 1) = p_s$ and $P(D = 0) = 1 - p_s$ for $0 \leq p_s \leq 1$; and $*$ defines an operator that performs the shortcut connection, given the value of D. The conventional dropout probability for hidden units is denoted as p_h.

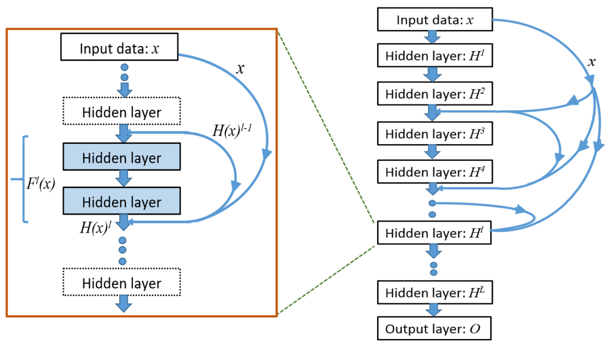

Fig. 1. Left: A hypothetical block of two hidden layers from the S-ResNet. Right: The whole S-ResNet [8]

2.2. Problem statement

Very deep networks have large expressive power for representing complex and compositional target functions. However, very deep networks learned with ReLUs can suffer reduction of representation capability as a result of dead ReLUs that go into saturation and blocking error gradients for updating associated model parameters during training. As such, these dead units fail to learn and therefore impact the learning capability of the model. This problem can be more severe in very deep networks (i.e. over 20 layers) where layer units much earlier in the model (i.e closer to the input) rely on the back propagation of error gradients over several hidden layers away. Furthermore, a deep network trained with dropout can be seen as an implicit ensemble model with shared parameters. At test time, model parameters are usually approximated by scaling them [18]. In [11], such approximation was shown to be more inexact for nonlinear networks. For very deep networks that compose ReLUs and trained with dropout, implicit model averaging is more significant and therefore the approximation of model parameters at test time could be more inexact than in shallower networks. Consequently, the aforementioned problems can work together to hurt model performance at test time.

3. PROPOSED: MAXOUT S-RESNET WITH ELASTIC NET REGULARIZATION

In this section, we present the details of our proposal for addressing the problems mentioned in Section 2.2. Particularly, our approach relies on learning maxout units, elastic net regularization and feature standardization as discussed below.

3.1. Learning units activation function via maxout

Conventional units in neural networks use 'hand-crafted' activation functions such as the rectified linear function, log-sigmoid function, tan-sigmoid, exponential linear function, etc. Maxout is an approach proposed in [11], where units activation functions are learned via the combination of piecewise linear functions. Moreover, [11] showed that maxout units allow a better approximation at test time of weights learned using the dropout technique [18], since model training with dropout can be considered as some sort of ensemble model that implicitly share model parameters. The output of a maxout unit k at layer l, $h(x)^l_k$, can be written as follows

$$h(x)^l_k = \max_{j \in [1,c]} o(x)^l_{kj}, \tag{3}$$

where $o(x)^l_{kj}$ is the output of a linear regressor j at layer l, and c is the number of feature extractors or channels across which we max pool.

3.2. Elastic Net Regularization (ENR)

Elastic Net Regulation (ENR) allows the explicit regularization of neural network models by penalizing model parameters. The elastic net regularization can be seen as a linear combination of $L1$-norm and $L2$-norm regularizations [19]. While the $L1$-norm (or LASSO) regularization results in strictly sparse solutions performing what can be seen as feature selection, the $L2$-norm (weight decay) regularization only encourages solutions with small values without indeed setting them to zero. The $L1$-norm regularization has the problem of discarding features when employed for correlated features [19]; the ENR aims to overcome this shortcoming by adding $L2$-norm to counteract the aforementioned problem with $L1$-norm regularization. For training neural networks with ENR, we can minimize the negative conditional log-likelihood of data given the model parameters along with the $L1$-norm and $L2$-norm penalties as follows

$$\begin{aligned} J(w) = & -\arg\min_{w} \sum_{n=1}^{N} \log P(y^{(n)}|x^{(n)}; W) + \lambda_1 \|W\|_1 \\ & + \lambda_2 \|W\|_2^2, \end{aligned} \tag{4}$$

where $J(w)$ is the cost function, $x \in R^d$ is the input data, $y \in R^k$ is the output, $W \in R^{d \times k}$ is the model parameter, n is the index of training samples, λ_1 and λ_2 control the magnitude of the $L1$-norm and the $L2$-norm penalties, respectively.

3.3. Feature standardization

Feature standardization (FS) (or Z-score normalization) in machine learning has been shown to improve generalization [20]. Features are rescaled to realize the characteristics of a standard normal distribution; that is, centering data to have a mean of zero and standard deviation of one. An explanation for the impact of feature standardization is that all features are transformed to the same scale such that features which vary less are not dominated by features that vary more. Feature

Algorithm 1: ENR S-ResNet+FS+SVM

```
1. Set $P_s$, $P_h$, $\lambda_1$, $\lambda_2$, $\eta$, $\alpha$ & $N_e$ using a validation set
2. Train S-ResNet via SGD+BN: feed data in mini-batches
3. If stopping condition for (2) is reached, discard the fully
   connected layers and do (4), else do (2)
4. Save model parameters
5. Forward propagate input data through model
6. Standardize the features obtained from the model's last layer
7. Train a linear SVM on the standardized features
```

standardization can be obtained by using

$$x_z^{(i)} = \frac{x^{(i)} - \bar{x}^{(i)}}{\sqrt{Var[x^{(i)}]}}, \tag{5}$$

where $x^{(i)}$ is an input data feature with index i, $\bar{x}^{(i)}$ is the mean and $Var[x^{(i)}]$ is the variance.
In this paper, we propose S-ResNet with maxout units for: (1) preserving the representation capacity of the S-ResNet, since ReLUs can die and learn nothing during training (2) improving parameters approximation of dropout trained S-ResNet at test time, since maxout units result in piecewise linear components for constructing units activation functions (3) tackle model overfitting with ENR as a result of the increase in model parameters introduced by the maxout units. Interestingly, we show that the features learned by the modified model using our training approach are very linearly separable such that a linear support vector machine (SVM) can replace the fully connected layers of the S-ResNet and achieve state-of-the-art results. Finally, we are able to improve experiment results by standardizing the features learned from the convolution layers stage of the maxout S-ResNet for training the linear SVM.
The training approach for the model that we propose in this paper is given as algorithm 1, where P_s, P_h, λ_1, λ_2, η, α and N_e are the dropout rate for the identity stochastic input shortcut connections, drop rate of hidden units, weight for $L1$-norm penalty, weight for $L2$-norm penalty, learning rate, momentum rate and maximum number of training epochs, respectively; SGD is stochastic gradient descent and BN is batch normalization. The aforementioned parameters are usually chosen using a validation set.

4. EXPERIMENTS

For the purpose of validating the proposed model, we use the popular USPS[1] and MNIST[2] datasets; these datasets are well known for benchmarking classification algorithms. The USPS dataset contains handwritten digits (0-9); for training and testing, there are 7,291 and 2,007 samples, respectively. The MNIST dataset contains handwritten digits (0-9); for

[1]http://www.cad.zju.edu.cn/home/dengcai/Data/MLData.html
[2]http://yann.lecun.com/exdb/mnist/

training and testing, there are 60,000 and 10,000 samples, respectively. The last 1,000 and 5,000 samples in the USPS and MNIST datasets are used for obtaining the hyper-parameters for the respective models. Since, we aim to demonstrate improved results in comparison to the earlier work [8], we train models of similar architectures in this paper. Particularly, we train 54-hidden layer maxout S-ResNets on the USPS and MNIST datasets using algorithm 1 presented in Section 3. The training details for both datasets are given below.

For the USPS dataset, we use $P_s = 0.8$, $P_h = 0.7$, $\lambda_1 = 10^{-5}$, $\lambda_2 = 10^{-3}$, $\eta = 0.1$, $\alpha = 0.9$ and $N_e = 100$. The learning rate was annealed from its initial value to the final value of 5×10^{-5} for the last 20 epochs. All convolutions in the maxout S-ResNet are of size 3×3 with zero padding; we applied max-pooling of size 2×2 twice. The model has 169K trainable parameters.

For the MNIST dataset, we use $P_s = 0.8$, $P_h = 0.8$, $\lambda_1 = 10^{-6}$, $\lambda_2 = 10^{-4}$, $\eta = 0.1$, $\alpha = 0.9$ and $N_e = 550$; the learning rate was annealed from its initial value to the final value of 5×10^{-5} for the last 270 epochs. All convolutions in the S-ResNet are of size 5×5 with zero padding; we applied max-pooling of size 2×2 twice. The model has 527K trainable parameters

Table 1 shows that our proposed maxout S-ResNet models trained on the USPS dataset outperform all models reported in earlier works. More interesting is that our proposed model outperforms the human classification error of 2.5% [13] and works that employed data augmentation (i.e. abbreviated as "aug.") for training. Furthermore, we observe that without feature standardization, the obtained test error for the maxout S-ResNet is 2.34%; this error rate is lower than the baseline result of 2.69% [8] and several other results from earlier works; see table 1. Table 2 shows the results for the trained maxout S-ResNets on the MNIST dataset, along with the state-of-the-art results. We note that [4] obtained an error rate of 0.21% with data augmentation. Conversely, our proposed models without data augmentation give very competitive results. Also, we observe that without feature standardization, the obtained test error for the maxout S-ResNet is 0.4%. Again, this error rate is lower than the baseline model result of 0.52% and several results from earlier works; see table 2. The confusion matrices of the maxout S-ResNet models reported in Table 1 and Table 2 are shown in Fig. 2(a) and Fig. 2(b) for the USPS and MNIST datasets, respectively.

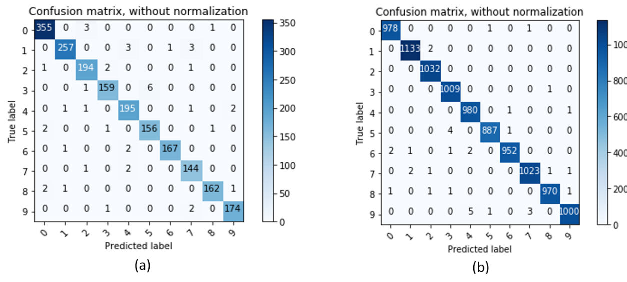

Fig. 2. Confusion matrix (a) USPS dataset (b) MNIST dataset

Models	Test error
Invariant vector supports [21]	3.00
Neural network (LeNet) [12]	4.20
Neural network + boosting + data aug. [12]	2.60
Manifold constraint transfer (MCT) [22]	2.99
Evolutionary compact embedding (ECE) [23]	3.90
Polynomial kernel SVM [24]	3.20
Tangent distance + data aug. [13]	2.50
Human performance [13]	2.50
Nearest neighbour [25]	5.60
Residual network (ResNet) - 54 hidden layers [8]	3.34
Baseline: 54 hidden layers S-ResNet [8]	2.69
Ours: 54 layers Maxout S-ResNet+ENR+SVM	**2.34**
Ours: 54 layers Maxout S-ResNet+ENR+FS+SVM	**2.19**

Table 1. Error rate (%) on the USPS dataset

Models	Test error
Polynomial kernel SVM [24]	0.56
Highway net-32 [7]	0.45
Maxout net [11]	0.45
Deep fried convet [26]	0.71
PCANet [27]	0.62
Network in network (NIN) [14]	0.47
Deeply supervised Network (DSN) [28]	0.39
ConvNet + L-BFGS [29]	0.69
Neural network ensemble + DropConnect [4]	0.52
Neural network ensemble + DropConnect + data aug. [4]	0.21
Stochastic pooling [15]	0.47
Residual network (Resnet) - 54 hidden layers [8]	0.76
Baseline: 54 hidden layers S-ResNet [8]	0.52
Ours: 54 layers Maxout S-ResNet+ENR+SVM	**0.40**
Ours: 54 layers Maxout S-ResNet+ENR+FS+SVM	**0.36**

Table 2. Error rate (%) on the MNIST dataset

5. CONCLUSION

Very deep networks have large representational capacity; hence, they require appropriate regularization to alleviate overfitting the training data. We build on an earlier work that employed residual learning with stochastic input shortcut connections for improving generalization of very deep networks (i.e. S-ResNet). In this paper, we propose to learn the activation function of units in the S-ResNet via maxout units since ReLU units can die out in training and impact representation capacity of very deep networks. We employ ENR for further regularization of the S-ResNet model due to increased parameterization based on the maxout units. Our experiments show that we outperform all earlier reported results, including human performance on the USPS dataset. On the MNIST dataset, we obtain very competitive results.

6. REFERENCES

[1] G. M. Y. Dauphin, X. Glorot, S. Rifai, Y. Bengio, I. Goodfellow, E. Lavoie, X. Muller, G. Desjardins, D. Warde-Farley, P. Vincent *et al.*, "Unsupervised and transfer learning challenge: a deep learning approach," in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, 2012, pp. 97–110.

[2] O. Delalleau and Y. Bengio, "Shallow vs. deep sum-product networks," in *Advances in Neural Information Processing Systems*, 2011, pp. 666–674.

[3] Y. Sun, X. Wang, and X. Tang, "Hybrid deep learning for face verification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 1489–1496.

[4] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proceedings of the 30th international conference on machine learning (ICML-13)*, 2013, pp. 1058–1066.

[5] M. Telgarsky, "Benefits of depth in neural networks," *arXiv preprint arXiv:1602.04485*, 2016.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[7] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Advances in neural information processing systems*, 2015, pp. 2377–2385.

[8] O. K. Oyedotun, A. E. R. Shabayek, D. Aouada, and B. Ottersten, "Training very deep networks via residual learning with stochastic input shortcut connections," in *International Conference on Neural Information Processing*. Springer, 2017, pp. 23–33.

[9] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *European Conference on Computer Vision*. Springer, 2016, pp. 646–661.

[10] H. Zhao, F. Liu, L. Li, and C. Luo, "A novel softplus linear unit for deep convolutional neural networks," *Applied Intelligence*, pp. 1–14, 2017.

[11] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," *arXiv preprint arXiv:1302.4389*, 2013.

[12] P. Simard, Y. LeCun, J. Denker, and B. Victorri, "Transformation invariance in pattern recognitiontangent distance and tangent propagation," *Neural networks: tricks of the trade*, pp. 549–550, 1998.

[13] P. Simard, Y. LeCun, and J. S. Denker, "Efficient pattern recognition using a new transformation distance," in *Advances in neural information processing systems*, 1993, pp. 50–58.

[14] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[15] M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," *arXiv preprint arXiv:1301.3557*, 2013.

[16] K. Greff, R. K. Srivastava, and J. Schmidhuber, "Highway and residual networks learn unrolled iterative estimation," *arXiv preprint arXiv:1612.07771*, 2016.

[17] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," *arXiv preprint arXiv:1609.04836*, 2016.

[18] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[19] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005.

[20] D. Bollegala, "Dynamic feature scaling for online learning of binary classifiers," *Knowledge-Based Systems*, vol. 129, pp. 97–105, 2017.

[21] B. Schölkopf, P. Simard, A. J. Smola, and V. Vapnik, "Prior knowledge in support vector kernels," in *Advances in neural information processing systems*, 1998, pp. 640–646.

[22] B. Zhang, A. Perina, V. Murino, and A. Del Bue, "Sparse representation classification with manifold constraints transfer," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4557–4565.

[23] L. Liu, L. Shao, and X. Li, "Evolutionary compact embedding for large-scale image classification," *Information Sciences*, vol. 316, pp. 567–581, 2015.

[24] S. Maji, A. C. Berg, and J. Malik, "Efficient classification for additive kernel svms," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 66–77, 2013.

[25] D. Keysers, J. Dahmen, T. Theiner, and H. Ney, "Experiments with an extended tangent distance," in *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, vol. 2. IEEE, 2000, pp. 38–42.

[26] Z. Yang, M. Moczulski, M. Denil, N. de Freitas, A. Smola, L. Song, and Z. Wang, "Deep fried convnets," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1476–1483.

[27] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma, "Pcanet: A simple deep learning baseline for image classification?" *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 5017–5032, 2015.

[28] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Artificial Intelligence and Statistics*, 2015, pp. 562–570.

[29] J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, Q. V. Le, and A. Y. Ng, "On optimization methods for deep learning," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 265–272.